CLAIMS

What is claimed is:

1. A method for displaying a structural view of a computer program during a debugging session, said method applying within a computer system, said method comprising:

displaying a portion of a program call graph (PCG), wherein said PCG includes a P_node symbolically representing a first procedure and a procedure relationship symbolically representing a calling association from said first procedure to a second procedure.

2. The method of claim 1, further comprising:

determining a condition for said first procedure while executing the computer program; and

marking said P_node based on said condition into a marked P_node, wherein said marked P_node is visually distinguishable from said P_node.

3. The method of claim 2, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

4. The method of claim 2, wherein said marking is changing a symbolic attribute from an unaltered P_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

5. The method of claim 1, further comprising:

creating a list of a plurality of PCG procedures from said portion of said PCG; and

recording said list of said plurality of PCG procedures onto a memory, said memory being retrievable.

6. A method for displaying a structural view of a computer program during a debugging session, said method applying within a computer system, said method comprising:

displaying a portion of a control flow graph (CFG), wherein said CFG includes a B_node symbolically representing a first basic block and a basic block relationship symbolically representing a calling association from said first basic block to a second basic block.

7. The method of claim 6, further comprising:

displaying within said B_node a line number associated with a source code statement of the computer program.

8. The method of claim 7, further comprising:

displaying within said B_node a portion of said source code statement.

9. The method of claim 8, wherein said portion of said source code statement can be alternately toggled for one of either displaying said source code statement or displaying said portion of said source code statement.

10. The method of claim 6, further comprising:

determining a condition for said first basic block while executing the computer program; and

marking said B_node based on said condition into a marked B_node, wherein said marked B_node is visually distinguishable from said B_node.

11. The method of claim 10, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

12. The method of claim 10, wherein said marking is changing a symbolic attribute from an unaltered B_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

13. The method of claim 6, further comprising:

creating a list of a plurality of CFG instructions from said portion of said CFG; and

recording said list of said plurality of CFG instructions onto a memory, said memory being retrievable.

14. A programmable storage device readable by a machine tangibly embodying a program of instructions executable by said machine to perform method steps for displaying a structural view of a computer program during a debugging session, said program applying within a computer system, said method steps comprising:

displaying a portion of a program call graph (PCG), wherein said PCG includes a P_node symbolically representing a first procedure and a procedure relationship symbolically representing a calling association from said first procedure to a second procedure.

15. The programmable storage device of claim 14, wherein said method steps further comprise:

determining a condition for said first procedure while executing the computer program; and

marking said P_node based on said condition into a marked P_node, wherein said marked P_node is visually distinguishable from said P_node.

16. The programmable storage device of claim 15, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

17. The programmable storage device of claim 15, wherein said marking is changing a symbolic attribute from an unaltered P_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

18. The programmable storage device of claim 14, wherein said method steps further comprise:

creating a list of a plurality of PCG procedures from said portion of said PCG; and

recording said list of said plurality of PCG procedures onto a memory, said memory being retrievable.

19. A programmable storage device readable by a machine tangibly embodying a program of instructions executable by said machine to perform method steps for displaying a structural view of a computer program during a debugging session, said program applying within a computer system, said method steps comprising:

displaying a portion of a control flow graph (CFG), wherein said CFG includes a B_node symbolically representing a first basic block and a basic block relationship symbolically representing a calling association from said first basic block to a second basic block.

20. The programmable storage device of claim 19, wherein said method steps further comprise:

displaying within said B_node a line number associated with a source code statement of the computer program.

21. The programmable storage device of claim 19, wherein said method steps further comprise:

determining a condition for said first basic block while executing the computer program; and

marking said B_node based on said condition into a marked B_node, wherein said marked B_node is visually distinguishable from said B_node.

22. The programmable storage device of claim 21, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

23. The programmable storage device of claim 22, wherein said marking is changing a symbolic attribute from an unaltered B_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

24. The programmable storage device of claim 19, further comprising:

creating a list of a plurality of CFG instructions from said portion of said CFG; and

recording said list of said plurality of CFG instructions onto a memory, said memory being retrievable.

25. A debugger for displaying a structural view of a computer program during a debugging session, said debugger operating within a computer system, said debugger comprising:

a displayer for displaying a portion of a program call graph (PCG), wherein said PCG includes a P_node symbolically representing a first procedure and a procedure relationship symbolically representing a calling association from said first procedure to a second procedure.

26. The debugger of claim 25, further comprising:

a condition determiner for determining a condition for said first procedure while executing the computer program; and

a marker for marking said P_node based on said condition into a marked P_node, wherein said marked P_node is visually distinguishable from said P_node.

27. The debugger of claim 26, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

28. The debugger of claim 26, wherein said marker changes a symbolic attribute from an unaltered P_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

29. The debugger of claim 25, further comprising:

a lister for producing a list of a plurality of PCG procedures from said portion of said PCG; and

a recorder for recording said list of said plurality of PCG procedures onto a memory, said memory being retrievable.

30. A debugger for displaying a structural view of a computer program during a debugging session, said debugger operating within a computer system, said debugger comprising:

a displayer for displaying a portion of a control flow graph (CFG), wherein said CFG includes a B-node symbolically representing a first basic block and a basic block relationship symbolically representing a calling association from said first basic block to a second basic block.

31. The debugger of claim 30, further comprising:

a line number displayer for displaying within said B-node a line number associated with a source code statement of the computer program.

32. The debugger of claim 31, further comprising:

a statement displayer for displaying within said B-node a portion of said source code statement.

33. The debugger of claim 32, wherein said portion of said source code statement can be alternately toggled for one of either displaying said source code statement or displaying said portion of said source code statement.

34. The debugger of claim 30, further comprising:

a condition determiner for determining a condition for said first basic block while executing the computer program; and

a marker for marking said B_node based on said condition into a marked B_node, wherein said marked B_node is visually distinguishable from said B_node.

35. The debugger of claim 34, wherein said marker changes a symbolic attribute from an unaltered B_node, wherein said symbolic attribute is taken from the group consisting of shade, highlight, color, border thickness, symbol size, symbol shape, and alternation of a visual characteristic.

36. The debugger of claim 34, wherein said condition is taken from the group consisting of an execution state, an execution frequency and an execution age, wherein said execution state corresponds to either said first procedure having been executed or said first procedure being nonexecuted, wherein said execution frequency is a rate of said first procedure being executed, and wherein said execution age is a time interval since said first procedure has been executed.

37. The debugger of claim 36, further comprising:

a lister for producing a list of a plurality of CFG instructions from said portion of said CFG; and

a recorder for recording said list of said plurality of CFG instructions onto a memory, said memory being retrievable.